MAGATARI™

VOLUME 1 NUMBER 8

for the ATARI 400/800

1	STAR TREK
2	THE DATABOSS #2
3	CLOCK
4	6502 TUTORIAL #2
5	3-D GRAPHING

$12.95 cassette, $17.95 disk

programmer's institute

a division of **FUTUREHOUSE™**

310½ West Franklin Street

P.O. Box 3470, Chapel Hill, NC 27514

1-800-334-SOFT

orderline

Editor's Notes

We have been informed by ATARI,Inc that the name of our magazine infringes on the ATARI name. Since it would be foolish for us to draw the ire of our best customer (they currently hold three subscriptions to MAGATARI), not to mention the fact that ATARI supplies the computers which we support with software,hardware,and computer furniture... we will henceforth be known as **MAGATAR.**

We hope you noticed the above references to hardware and computer furniture. We currently hold exclusive marketing rights to the **EDUMATE**^{tm} light pen. While our light pen costs only $29.95 and performs all the functions of any other light pen on the market for the ATARI or VIC,it costs $40.00 less than the next cheapest pen. We are also designing furniture for your ATARI 400/800. The two models currently in the design stage include an inexpensive model and a deluxe model. Both models will remove all those wires hanging out of your ATARI and peripherals.

You can see models of the furniture (also our light pen, and other lines of software) at the South East Computer Show in Atlanta,GA.Dec.9-12,1982. If any subscribers plan to be in the Atlanta area between the above dates, feel free to stop by and see our new catalog and product lines.

Don't forget to enter our two contests begining with this issue! Both are designed to improve the circulation of good programs from subscribers. Our goal is to have 2 more programs in each issue that are submitted by our subscribers. Prizes for programs which are accepted for publication in **MAGATAR** include $75 or a free years subscription to **MAGATAR.** Winners of our **MAGATAR MEDALLION AWARD** will receive prizes up to $250 plus royalties. So fill out those entry forms today!

STAFF OF MAGATAR

FUTUREHOUSE,INC.,PUBLISHER

ANDREW A. HOCK, EDITOR
CHRIS F. STEWART,MANAGING EDITOR
JAMES A. DOUGLAS,MARKETING DIRECTOR
JAN WEBSTER, EDITORIAL ASSISTANT
ANN CARPENTER, EDITORIAL ASSISTANT

ASSOCIATE EDITORS:
DARREN DeLOACH,
c/o MAGATAR,P.O.BOX 3191,
CHAPEL HILL,N.C.27514.

CHRIS AIKEN,
c/o MAGATAR,P.O.BOX 3191,
CHAPEL HILL,N.C.27514.

Loading Instructions

Here are the programs included in Volume 1 Number 8. For all cassette subscribers,the line counter numbers are listed to the right of the names of the programs:

PROGRAM NAMES	COUNTER #
THE DATABOSS^{tm}	10
UTILITY-SYSTEM CLOCK	177
STAR TREK	104
MACHINE LANGUAGE	197
3-D GRAPHING	256

Cassette users should advance their tape recorder to the specified line number,type in CLOAD,and press [RETURN] twice (the programs are saved on both sides of the tape).

All diskette subscribers should insert the MAGATAR diskette into their drive, and turn on their machine. If you wish to load the programs separately, do the following:

PROGRAM NAMES	INSTRUCTION
MENU PROGRAM	RUN"D:MAGATAR"
DATABOSS^{tm}	RUN"D:DATABOSS.2"
STAR TREK	RUN"D:STAR TREK"
MACHINE LANGUAGE	RUN"D:MACHINE.2"
UTILITY-CLOCK	RUN"D:CLOCK"
3-D GRAPHING	RUN"D:TDGRAPH"

THE DATABOSS^{tm} - Part 2

Last issue we presented a shell of a database program that allowed you to create, and edit a file. In this issue,we have added more editing commands, and a merge program that will allow you to merge any existing file with a **DATABOSS^{tm}** file, (will be available next issue).

DISK VERSION

The main menu of THE DATABOSS^{tm} looks like this:

```
                THE DATABOSS[tm]  [VS.1.1]
          copyright(c) 1982 by FUTUREHOUSE

[A] Create file          [B] Load file
[C] Select print         [D] File inquiry
[E] Edit sort links      [F] File merge
[G] Format disk          [H] Disk directory
[I] Edit file spec       [J] End this job

                What is your choice?
```

Options C,E, and I will be implemented in a future issue. Creating a file and loading a file are the same as last issue. For the benefit of those who did not receive the last issue,we have included the documentation for these options in this issue.

Upon selecting Option A (Create file),the following screen will appear:

THE DATABOSS[tm] - CREATE MODE

Press (M)enu or [RETURN] to continue

If you select (M) you'll return to the main menu. Otherwise the next screen appears:

In our example,we will create a home inventory file. The file will contain records, each with the following fields and field characteristics:

TITLE	LENGTH	TYPE
1.NAME OF ITEM	25 char	alphanumeric
2.CONDITION	2 char	alphanumeric
3.QUANTITY	3 char	numeric
4.PRICE PAID	10 char	dollars/cents
5.PRESENT VALUE	10 char	dollars/cents

after pressing the [RETURN] key, the next screen will appear:

THE DATABOSS[tm] - CREATE MODE

( ) number of fields per record

In our home inventory example there are 5 fields (you can have up to 99 fields), so press 5 then [RETURN].

THE DATABOSS[tm] - CREATE MODE

enter name of this application

The name of the application can be up to 25 characters long. You can call it anything you wish,but for the sake of clarity,we'll call it 'HOME INVENTORY FILE.' Type this in,then press [RETURN].

THE DATABOSS[tm] - CREATE MODE

enter name of this application

HOME INVENTORY FILE

enter filename for application

DISK FORMAT WILL BE "D:filename.DAT

The disk file name can be up to 8 characters long.We'll type in 'HOMEINVT'.

The keyboard will buzz,warning you that you are about to enter the create mode,and you will have two options.Key in [M] to return to the main menu(i.e. if you accidentally hit the wrong key),or hit [RETURN] to continue to the first step of creating a file

The next screen looks like this:

THE DATABOSS^{tm} - CREATE MODE

---

enter title for this field

The name of the first field in our example is 'NAME OF ITEM'. Type this in, then press [RETURN].

THE DATABOSS^{tm} - CREATE MODE

---

enter title for this field

NAME OF ITEM

---

enter length of this field (1-99)

( )

In our example,the lenght of this field was selected to be 25 characters, so key in 25.

THE DATABOSS^{tm} - CREATE MODE

---

enter title of this field

NAME OF ITEM

---

enter length of this field (1-99)

(25)

SELECT FIELD EDIT FORMAT:

1. alphanumeric 2. numeric
3. date mm/dd/yy 4. dollars/cents

( )

Since the NAME OF ITEM field should be able to hold any character (numbers or letters), select Option 1 (alphanumeric characters). You will then be given the following prompt:

ARE THESE ENTRIES CORRECT (Y/N) ( )

If you've made any errors, simply press 'N' and repeat the above process. If the entries are correct, press 'Y'.

Repeat the above process for the next four fields, using the features described for each field. After completing the process, the following screen will appear:

THE DATABOSS^{tm} - CREATE MODE

ENTER # OF RECORDS IN FILE

( )

Depending on the size of the record,you can have up to 999 records in a file. For our purposes, we'll create 200 records for our HOME INVENTORY FILE. Type in 200.

You will then be asked to:

INSERT DISKETTE IN DRIVE 1

Insert a formatted diskette in your drive (you can also format a diskette by selecting Option 5 from the main menu). Then press any key to start the create file process. The busy light on your drive will turn on, and you will be informed the program is NOW CREATING DATABASE FILE. After the process is complete, control will return to the main menu.

---

(1) THE DATABOSS^{tm} - CREATE MODE

---

enter title of this field

NAME OF ITEM

enter length of this field (1-99)
(25)

SELECT FIELD EDIT FORMAT

1.alphanumeric 2.numeric
3.date mm/dd/yy 4.dollars/cents
(1)

SELECT SORT OPTION FOR THIS FIELD

1.ascending 2.descending
3.no sort for this field

Select 1,since we'll want to keep this field sorted. The next prompt will appear. There is also an option now for a disk directory. Simply insert the disk you wish to see the directory for, and select Option H.

After loading or creating a file,select Option D to edit the file. In our example of a Home Inventory File, the followiing screen will appear:

THE DATABOSS^{tm} - FILE INQUIRY MODE

REC# ( ) FILE: HOME INVENTORY FILE

NAME OF ITEM

CONDITION QUANTITY

PRICE PAID PRESENT VALUE

[E]dit [M]enu [N]ext [L]ast [S]earch
[F]orward [B]ack [D]elete [R]ecord #

The DATABOSS^{tm} will still format screens for you, and displays the exact length assigned every field. To add a record, select [E] for EDIT. There is not an option to [A]dd or [C]hange. If you want to add a record, access the record you wish to add by using [N]ext or [L]ast to advance or go back a record at a time, or select [R] for record #, and enter the record number desired to access the record immediately. Then select [E] to enter the data.

If you make a mistake, or wish to change the data in a field, simply press [E]. If you do not want to change the field under the cursor, press [RETURN] without entering any data and the cursor will advance to the next field without changing the data in the previous field.

To delete a record, simply access the desired record, and press [D] for delete. The Option [F]orward and [B]ack are similar to [N]ext and [L]ast, but are used when there is more than one screen per record. With this new version of DATABOSS^{tm}, you can have as long a record as you want (up to 99 fields), and can also have up to ten disk of data.

If you wish to search the file for a particular record, select [S]earch. You will be asked the field number of the field you want to make a search, and then the data to search for. The search will begin, and stop at the first record that meets the criterion of the search. If you wish to continue the search, select [S]earch again and key [C] for continue to the field number prompt. When finished editing, select [M] to return to the main menu.

In the next issue, we will add a seperate program that will allow you to format reports, and sort.

UTILITY OF THE MONTH - System Clock

Our utility of the month is a machine language program that will display a clock on your screen. Better yet, you can display or erase the time by pressing the [OPTION] key. The principle behind the clock is the Vertical Blank Interrupt (VBI) discussed in our Machine Language Tuturial #2.

Here's how it works:

First, we have to come up with an algorithm that we can expand into 6502 machine code. Before we can implement the clock we have to write a BASIC program to implement the clock. We also need storage locations to hold each of the two digits for hours,minutes,seconds and 60th of a second. We also need a storage location that tells our routine whether it is AM or PM, and one that let's us know if the clock is active or inactive.

Once we've decided on all the storage addresses we need, we can jump into the routine.

```
0000          0190          *=   $0600
0600 D8       0200          CLD
0601 A200     0210          LDX  #$00
0603 FEF706   0220 TIMEL    INC  TIME,X
0606 BDF706   0230          LDA  TIME,X
0609 C90A     0240          CMP  #$0A
060B D05A     0250          BNE  PRINT
060D A900     0260          LDA  #$00
060F 9DF706   0270          STA  TIME,X
0612 E8       0280          INX
0613 FEF706   0290          INC  TIME,X
0616 BDF706   0300          LDA  TIME,X
0619 C906     0310          CMP  #$06
061B D04A     0320          BNE  PRINT
061D A900     0330          LDA  #$00
061F 9DF706   0340          STA  TIME,X
0622 E8       0350          INX
0623 E006     0360          CPX  #$06
0625 D0DC     0370          BNE  TIMEL
```

The following is a diagram showing the storage location addresses, and their position on page 6:

	NAME		HEX	DEC
	TEST		$06F3	1779
	TEMPORARY STORAGE LOCATIONS		$06F4	1780
			$06F5	1781
			$06F6	1782
	MSECS1		$06F7	1783
	MSECS2		$06F8	1784
	SECONDS1		$06F9	1785
TIME	SECONDS2		$06FA	1786
	MINUTE1		$06FB	1787
	MINUTE2		$06FC	1788
	HOURS1		$06FD	1789
	HOURS2		$06FE	1790
	AMPM		$06FF	1791

These MSECSI,SECONDSI, etc. are the least significant digits (L$D) of seconds/60,seconds, etc.. TIME is an index into the time addresses that we used above to increment each address in turn. If a LSD reaches $0A,HEX,10DEC; it is changed to 0 and it's most significant digit (MSD) is increment by 1. If a MSD reaches 6,it is changed to 0 and a higher LSD is incremented. After each increment, control is passed to the print section of the routine, and we print the new time.

In case you're wondering what the CLD command is at the beginning of the routine, it means Clear Decimal Arithmetic. In some operations, we must ensure that binary arithmetic is restored for our routine. We'll talk more of BCD Arithmetic in a future issue.

```
0627 EEFD06 0380         INC  HOURS1
062A EEF406 0390         INC  HOURS
062D ADF406 0400         LDA  HOURS
0630 C90A   0410         CMP  #$0A
0632 3033   0420         BMI  PRINT
0634 D00B   0430
0636 EEFE06 0440         INC  HOURS2
0639 A900   0450         LDA  #$00
063B 8DFD06 0460         STA  HOURS1
063E 4C6706 0470         JMP  PRINT
0641 C90C   0480 TEST12  CMP  #$0C
0643 D013   0490         BNE  TEST13
0645 ADFF06 0500         LDA  AMPM
0648 C901   0510         CMP  #$01
064A D006   0520         BNE  AM
064C CEFF06 0530         DEC  AMPM
064F 4C6706 0540         JMP  PRINT
```

```
0652 EEFF06 0550 AM      INC  AMPM
0655 4C6706 0560         JMP  PRINT
0658 C90D   0570 TEST13  CMP  #$0D
065A D00B   0580
065C A901   0590         LDA  #$01
065E 8DFD06 0600         STA  HOURS1
0661 8DF406 0610         STA  HOURS
0664 CEFE06 0620         DEC  HOURS2
```

If it is time for an hour to be incremented, we continue down the routine and increment the LSD of hours. We compare HOURS to OA,HEX;10,DEC. If the comparison is greater (not minus and not equal) we branch to TEST12. If it's equal we increment the MSD of hours, change the LSD to 0, and jump to PRINT.

If it's equal to 12, we must change AM to PM, or PM to AM. If it's equal to 13, we must change the LSD of hours to 1 and MSD to 0.

```
0667 A558   0630 PRINT    LDA  $58
0669 85CF   0640          STA  $CF
066B A559   0650          LDA  $59
066D 85D0   0660          STA  $D0
066F A00D   0670          LDY  #$0D
0671 A206   0680          LDX  #$06
0673 AD1FD0 0690          LDA  OPTION
0676 C903   0700          CMP  #$03
0678 D012   0710          BNE  NOOPTION
067A ADF306 0720          LDA  TEST
067D C900   0730          CMP  #$00
067F D008   0740          BNE  RESTART
0681 A901   0750          LDA  #$01
0683 8DF306 0760          STA  TEST
0686 4C62E4 0770          JMP  $E462
0689 CEF306 0780 RESTART  DEC  TEST
068C ADF306 0790 NOOPTION LDA  TEST
068F F034   0800          BEQ  JUMP
```

Here we start to print the clock, but only if it's active. We store the location of screen ram ($58 and $59),to another page 0 location and checks to see if the (OPTION) key is being pressed. If it's not pressed we branch to NOOPTION. If it is, we check the TEST address to see if the clock is active. If it is we deactivate the clock and jump out of the routine. If it's not,we reactivate it.

If we branched to NOOPTION we check the TEST address. If its not active we jump to the VBI routine in ROM,otherwise we start printing the clock.

```
0691 BDF706 0810 LPRINT LDA  TIME,X
0694 8DF506 0820        STA  NUMBER1
0697 E8     0830        INX
0698 BDF706 0840        LDA  TIME,X
069B 8DF606 0850        STA  NUMBER2
069E 20C806 0860        JSR  SEPRAT
06A1 CA     0870        DEX
06A2 CA     0880        DEX
06A3 CA     0890        DEX
06A4 E000   0900        CPX  #$00
06A6 3003   0910        BMI  FIN
06A8 4C9106 0920        JMP  LPRINT
```

Here we use the TIME index in the other directry from hours to seconds/60 instead of MSECS to HOURS, so we can print the clock with hours on the left and seconds/60 is on the right.

```
06AB A980    0930 FIN  LDA  #BLANK
06AD 91CF    0940      STA  ($CF),Y
06AF C8      0950      INY
06B0 ADFF06  0960      LDA  AMPM
06B3 C901    0970      CMP  #$01
06B5 D005    0980      BNE  AM1
06B7 A9B0    0990      LDA  #PLET
06B9 4CBE06  1000      JMP  PM1
06BC A9A1    1010 AM1  LDA  #ALET
06BE 91CF    1020 PM1  STA  ($CF),Y
06C0 C8      1030      INY
06C1 A9AD    1040      LDA  #MLET
06C3 91CF    1050      STA  ($CF),Y
06C5 4C62E4  1060 JUMP JMP  $E462
```

After we print the clock, we print a blank space, and then test AMPM to see if we should print an "A" or a "P". After deciding this and printing it, we print an "M" and then jump out to the V Blank Routine.

```
06C8 ADF606  1070 SEPRAT LDA  NUMBER2
06CB 18      1080        CLC
06CC 6990    1090        ADC  #$90
06CE 91CF    1100        STA  ($CF),Y
06D0 C8      1110        INY
06D1 ADF506  1120        LDA  NUMBER1
06D4 18      1130        CLC
06D5 6990    1140        ADC  #$90
06D7 91CF    1150        STA  ($CF),Y
06D9 C8      1160        INY
06DA C018    1170        CPY  #$18
06DC F005    1180        BEQ  RETURN
06DE A99A    1190        LDA  #COLON
06E0 91CF    1200        STA  ($CF),Y
06E2 C8      1210        INY
06E3 60      1220 RETURN RTS
06E4 68      1230        PLA
06E5 A907    1240        LDA  #$07
06E7 A206    1250        LDX  #$06
06E9 A000    1260        LDY  #$00
06EB 205CE4  1270        JSR  $E45C
06EE 60      1280        RTS
06EF         1290        ,END
```

Here is the subroutine used to print the clock to the screen. First the MSD is printed, then the LSD is printed for each time part. Then a colon is printed. This subroutine is repeated for the hours,minutes,seconds, and seconds/60.

The routine starting at $06E4 HEX is the routine that sets the VBI rector to point at page 6. See the discussion in our Machine Language Tutorial #2 for more details.

The map is composed of 900 sectors, arranged in a 30 by 30 matrix. The sectors are numbered 0 to 29 both horizontally and vertically. Thus, the upper left corner of the map is sector 0,0, the upper right corner is 0,29, the lower left is 29,0, and the lower right is 29,29.

The map is randomly populated with Klingon warships, stars, planets (either hostile, neutral, or friendly), black holes, and 5 Starbases, one always being placed at 15,15 and the other 4 at random locations.

After typing RUN, about 30 seconds will pass, and the prompt SHIP'S NAME?, will be printed. Answer this with a 10 character or less name of your ship, such as Enterprise or Discovery. The next prompt is SHIELDS?, and needs some discussion.

Your ship has a total of 9999 energy units available. These are distributed between shields and energy used for engines and phasers. Thus, if you enter 3000 (typical) here, you'll have 6999 units left to move and shoot with. Your status display will constantly monitor both quantities. The sum total of the two will never exceed 9999, but can always be much less. If at any time you run out of energy (not counting shields!), your ship will die. You can replenish energy by docking with a starbase at any time (up to 9999 - shields). Note that it crosses 10 units per turn if your condition is green, while red alert costs 30 units, thus even while doing nothing you expend energy.

After entering your shield energy (2000-3000 suggested), the screen clears and you see:- a) a visual scan of your sector and up to 5 sectors in all directions away from you. b) either a status reading or (more likely) a damage report (which will all be zeros). While on red alert, which occurs if there are any Klingons on your scan, you will always get a damage report and a science computer scan.

First let's discuss the damage report. You will see all zeros right now, but if at any time you lose all your shield energy, you get damage numbers. If any are greater than 10, that function is damaged and unavailable. Hit [space] or [return] to exit the damage report.

Next comes the science computer scan. It will describe the contents of your sector map. The scan runs horizontally then vertically, (ie. it scans all x-values of each y-value, from top to bottom). It explains the alliance of planets (either friendly,neutral, or hostile), and the energy and damage levels of all Klingons. Press [space] to continue the scan, or [return] to exit the scan.

Now let's look at the sector map. It shows the locations of all stars, planets, Klingons, starbases, black holes, and your ship. Here are the symbols:

. - empty sector
[] - Klingon
* - star
@ - planet
  - starbase
- black hole

You are represented by the first 2 letters of your ship's name.

After the science scan (if you are on red alert, immediately upon getting the sector map if not), you will get your status report. It

includes:
Energy - as previously discussed, energy available for movement and phasers.
Sector - your location (X,Y) on the map.
Torpedoes - number of torpedoes left (max 3). Set to 3 every time you dock.
Condition - Red if any Klingons are present Green if not.
Shields - energy left in shields.
Stardate - unit of time in game, starts at 2000.
Score - present score you have achieved.

Now you'll see the COMMAND? prompt.
There are 8 possible commands, which are:
[S]hields - reset shield energy. Simply answer the prompt with the amount you want to set them to, if you are reducing them, energy is credited to your total energy.

[D]amage - gives you the damage report (if available). Press [space] or [return] to exit.

[R]epair - repairs up to 75% of damage from each function, and adds 200 units of energy (not to exceed 9999-shields). Every time you dock, 50% repair is performed as well.

[P]hasers - answer the first prompt with the sector (X,Y) of your target. A message describing the target will appear. Answer ENERGY? with the amount of energy you wish to fire, or 0 if you don't want to fire. You may fire only at a sector which shows up on your sector map.

[T]orpedo - prompt is DIRECTION?. Torpedoes fire only in straight lines and stop at the first thing they hit, (phasers shoot in any direction and strike only the target sector). Torpedoes are automatic kills on Klingons or planets, destruction of a planet, however, results in a loss of score. Stars or black holes stop torpedoes. Torpedoes travel 10 sectors, and therefore may strike something not on your display.

```
     2
   3   1
 4   :   0 - Directions
   5   7     for torpedoes.
     6
```

[M]ovement - enter target sector (anywhere on map), then speed. The faster you go, the less time it takes, but it also takes more energy. You may die if you try to go too far too fast, (from running out of energy). If you try to land on an occupied sector, the computer places you on some nearby empty sector. If you land adjacent to a hostile or neutral planet, it may attack you, possibly even killing you!

[C]omputers - answering the prompt with a 1 gives you the standard science computer scan, a 2 gives you the ship's computer scan, which is a sector map scan, from every starbase (15,15 first). Thus you know where to land, or the location of any starbase. Press [space] to get the next starbase sector scan, or [return] to exit this command.

[Q]uit - to end the game. It will give you:
Total stardates - how long your game lasted.
Score - total score.
killed - number of Klingons killed in game.
killed/stardate - average number of Klingons killed per stardate.

Note that after commands S,D, and C you return to the command loop, while R,P,T, and M initiate Klingon battle logic.

Some notes on Klingon logic - a Klingon does only one of two things: move or fire. Movement costs no energy, it shoots, only up to its total energy. It can be either damaged or lose energy by your fire CDA and E from science scan. If E[0 or DA]25 the Klingon dies.

The game may continue infinitely, as every 5 stardates 0 - 5 new Klingons enter the map. It is conceivable to kill all of them, but unlikely.

Please note that whenever you move, the Klingon ships repair themselves. Thus, it may be to your advantage to kill a damaged Klingon before you dock, or move somewhere else.

So - play the game, and have fun!

Program by Darren DeLoach.

3 - D GRAPHING

This program graphs an equation for you in 3-D, rotates it at another angle and graphs it again. There is also a subrouting furnished and explained so you can graph any equation on any rotation.

Our machine language tutorials will combine text in our magazine, and examples from programs on cassette or diskette. Before reading further, load and run the 6502 Tutorial Program.

In issue #7 we talked about the structure of the 6502 chip... the Accumulator, X-Register, Y-Register, Program Counter, Stack, and the Processor Status Register. We also showed you some simple machine language routines such as adding two numbers, and printing to the screen. Now you're ready to learn about the Operating System.

At one time or another, you probably wondered why you needed the 10k Operating System Cartridge, and a Basic Cartridge plugged into your ATARI at the same time. Well, the Basic Cartridge is simply one way of communicating with the OS, which in turn tells the 6502 what to do. In most other microcomputers, all the OS has to do is keep track of the microprocessor. However, in the ATARI, the OS not only has to control the 6502.... it also has to preside over the ANTIC, GTIA, POKEY, and PIA chips.

ANTIC's job is to get data from memory, which is used for screen display. In other words ANTIC controls playfield and player - missile graphics. While ANTIC controls the graphics display, GTIA prints the information to the TV or monitor. It also controls color-luminance of characters and players, and the priority of overlapping objects on the screen.

POKEY controls the sound registers, and all the other noises you hear from your ATARI. PIA controls communication between the computer and any peripherals that may be attached to it.

Now that you can appreciate all the work the OS is doing, let's see what makes the OS work. First, ther has to be a routine that is executed whenever the ATARI is turned on (called COLDSTART). This routine allows the OS to begin communication with all the other chips in the system to determine whether a disk drive (or drives) is connected and operational, (or a cassette recorder), whether a cartridge is marked in the left slot, and if one is placed in the right slot. All you'll ever see of COLDSTART is the READY sign that appears on the screen a couple of seconds after you turn on your ATARI. The routine that is executed when you press [SYSTEM RESET] (called WARMSTART) is very similar to the COLDSTART routine. The above routines are called the OS Monitor (since they monitor the OS communication between all the chips). The OS also has a ROM character set, a FLOATING POINT routine which allows mathematical functions such as +,-,*,/, as well as exponential and logarithmic functions, software and hardware timers, interrupt processing routines, and OS System Vectors.

What does all this stuff have to do with programming? What is an Interrupt, and what are OS System Vectors? Well, ther are 2 kinds of interrupts which we'll concern ourselves.... Non-Maskable Interrupts (NMI) and Interrupt Requests (IRQ). An IRQ happens everytime you press a key. The ATARI may be busy updating a timer or displaying the screen, but as soon as you press a key, it stops (interrupts!) everything it's doing, and processes the key pressed. If no program is running, or if the interrupt is in conjunction with an INPUT statement, it will also print the character to the display, and advance the cursor position. However, an IRQ is only an Interrupt Request. It has a lower priority than an NMI. In other words, if an NMI is taking place, the IRQ must wait until the NMI has returned control to the OS....only then can a key pressed be processed. This explains why when a program is loading from a cassette or disk, no key will print to the screen until the loading process is complete. Only upon completion will the key you pressed while the program was loading, print to the screen.

What is an NMI? There are 3 types of NMI's....[SYSTEM RESET] (WARMSTART), Vertical Blank Interrupt, and Display List Interrupt.

Your ATARI displays the screen on your TV set using scan lines. There are 192 scan lines to each TV screen that are printed from the top to the bottom of the screen. This precess is performed 60 times per second, and every time the 192nd scan line has completed printing, before it starts to print the 1st scan line again, a VBI is executed. Typically, the VBI updates system timers, and other screen oriented data. In your ATARI, the VBI also updates a software vector called ATTRACT, which resides at 77 DEC, 4D HEX, and makes your TV start changing colors after you've left it on for 9 minutes or so without pressing a key. (Enabling an IRQ, by the way, resets this address to 0).

There just happens to be an OS RAM vector at 546,547 DEC;$223 HGX which you can change to perform your own routines before continuing on to the VBI routine in ROM.

A DLI is different from a VBI in that a DLI can be performed before each mode line of the display. Every time you as a user, enters a graphics mode, ANTIC builds a Display List explaining to GTIA exactly what mode it's printing on. The first three mode lines of any Display List are blank to allow for overscan on some TV's.

The next 3 bytes contain the Load Memory Scan instructions. The first byte tells GTIA the mode to print the first line, and the next two bytes contain the address where screen RAM begins. The next set of addresses contain the ANTIC mode that is to be displayed. For example, in GRAPHICS 0 (ANTIC mode 2) there are 24 mode lines. The first mode line is included in the LMS instruction, and the other 23 are located in the 23 locations following the LMS instruction. Then there is an ANTIC instruction which forces ANTIC to wait for the VBI, and then jumps to the begining of the DL again. The DL for GR.0 is only 32 bytes long and would look something like this (this DL assumes a 48k machine).

```
ADDRESS              VALUE (DEC)
39968                 112
   69                 112  Blank 3*8 scan lines
   70                 112
   71                  66  Display ANTIC mode 2
   72                  64  = 156x256+64=40000 = Begining
   73                 156  of screen RAM
   74                   2
   75                   2
   76                   2
   77                   2  Display another 23 mode
   78                   2  lines of ANTIC mode 2
   79                   2
   80                   2
   81                   2
   82                   2
   83                   2
   84                   2
   85                   2
   86                   2
   87                   2
   88                   2
   89                   2
   90                   2
   91                   2
   92                   2
   93                   2
   94                   2
   95                   2
   96                   2
   97                  65  Wait for VBI, then jump
   98                  32  to 156x256+32=39968 (back
   99                 156  to begining of DL).
40000  Begining of screen RAM.
```

The WARMSTART routine is boring, so we won't talk about it except to say that location 580 DEC, $244 HEX is the software OS vector that determines whether COLDSTART (machine turned on) or WARMSTART (normal [SYSTEM RESET] pressed) is performed upon pressing the [SYSTEM RESET] key. If you POKE 580,1 and press [SYSTEM RESET], your ATARI will think you have just turned on the machine and attempt to reboot.

Let's continue on with our 6502 Tutorial Program, and see how we can apply the OS RAM vectors to our own use.

Display Lists can be a handy tool. They allow us to mix graphic modes on the same screen. We acn also create a DLI in any mode line

by adding 128 to the ANTIC mode number. For instance, in our previous example we showed you a DL for GRAPHICS 0 (ANTIC mode 2). If we add 128 to a mode line instruction we can change the DLI routine to point to our own routine before it prints that mode line. All we have to do is write a routine change the DLI vector to point to our routine, and include a Return from Interrupt (RTI) Instruction at the end of our routine. Let's look at an example in the 6502 Tutorial program.

Here's the 6502 program that we included in the DLI routine:

```
0000       10              *= $0600
00CE       20   TEST         = $CE
D018       30   COLOR 2      = $0018
D40A       40   WSYNC        = $D40A
00CF       50   ST1          = $CF
00D0       60   ST2          = $D0
0600 48    70   PHA
```

When adding any routine to a DLI, we must save any registers we use in our routine, before we begin. Since we only use the Accumulator in ours, it's the only register we pushed onto the stack. TEST is an address that will contain a 0 or a 1. It will hold a 0 if we are stuffing a new color into a mode line, and a 1 if we are stuffing the original color in a mode line. (This is why we needed to add 128 to 2 consecutive DL Instruction). COLOR 2 is the hardware register for Color Register 2. We cannot use the RAM vector for CR 2 (710 DEC,$2C6 HEX) because it is updated on the VBI routine (when the screen has finished printing). By stuffing the color into the hardware register, the change is instantaneous. WSYNC is a register in ROM that when addressed with any value freezes the action of the 6502 chip until a scan line has finished printing. If we did not address WSYNC, the line would look uneven and messy, since we never can precisely time the stuffing of the color register with the printing of scan lines within a mode line. ST1 and ST2 hold the color to be stuffed into the mode line, and the original color, respectively.

```
0601   ASCE     80    LDA    TEST
0603   D00D     90    BNE    ONE
0605   A5CF    100    LDA    ST1
0607   8D0AD4  105    STA    WSYNC
060A   8D18D0  110    STA    COLOR 2
060D   E6CE    120    INC    TEST
060F   4C1C06  130    JMP    EXIT
```

First, we compare test to 0. If it's not 0, it must be one, and we brach to that routine. Otherwise, we load the accumulater with the color we want to stuff into the mode line, and store it in WSYNC (to address it), and in the hardware register (COLOR 2). Then we increase the volume in TEST from 0 to 1 and jump to exit the routine.

```
0612   C6CE    140   ONE    DEC    TEST
0614   A5D0    150          LDA    ST2
0616   8D0AD4  155          STA    WSYNC
0619   8D18D0  160          STA    COLOR2
```

If the value in test is equal to 1, we decrement TEST back to 0, and load the accumulator with the original color. As before, we store this WSYNC and COLOR2.

```
061C   68   170   EXIT    PLA
061D   40   180           RTI
```

Here we restore the value that was in the accumulator before we started our DLI routine, by pulling the value back off the stack. We return from the interrupt and we're done.

If you continue on to the next page of our 6502 Tutorial Program, you'll notice that sometimes the whole bottom of the screen turns black instead of one mode line. This is because we are POKEING new values (2 or 130) into the DL instructions from BASIC, which is too slow to keep up with DLI's and VBI's. A 6502 subroutine which changes the DL Instructions for us would solve this problem.... why don't you try to write it? (we'll give you the answer in Issue # 9).

Next month we'll show you how to use the horizontal and vertical scroll registers to do both coarse and fine scrolling. Until then, the staff at **MAGATAR** wishes you a **MERRY CHRISTMAS** and a **HAPPY NEW YEAR!**